The purpose of this lab was to use the accelerometer in an Android smartphone, get introduced to Java, and revise code for displaying the data. An application was created to display the phone’s accelerometer data on the screen. The provided Java code was revised to include both the maximum acceleration recorded and the current acceleration values. First revised sections of code are presented. Then screenshots of the application running on my Droid 3 phone are presented to show the functionality of the sensor and the application.

The following sections of code are from MainActivity.java, main.xml, and strings.xml, respectively. The first section of edited code, which comes from MainActivity.java, adds the variables for maximum acceleration and decides the values that will be sent for display. The first block for variable declaration is included at the top of the code with other variable declarations.

```
FROM MainActivity.java:
/* Three variables are created for storing maximum acceleration values and initialized at zero. They are private because they are only needed in this package. They are float variables to match the output of the sensor.*/
private float maxX = (float)0.0;
private float maxY = (float)0.0;
private float maxZ = (float)0.0;
```

The next block is added to the @Override for the onSensorChanged() method, just after the previous instance of TextView class. It is responsible for calling the identifier name given to the view in the main.xml code. These instances are later given strings of values to display on the phone’s screen.

```
/* The following code assigns six instances of the TextView class for displaying the acceleration values.*/
TextView tvmX= (TextView)findViewById(R.id.x_max);
TextView tvmY= (TextView)findViewById(R.id.y_max);
TextView tvmZ= (TextView)findViewById(R.id.z_max);
TextView tvlX= (TextView)findViewById(R.id.x_last);
TextView tvlY= (TextView)findViewById(R.id.y_last);
TextView tvlZ= (TextView)findViewById(R.id.z_last);
```

The if-statements below are responsible for recording the maximum accelerometer values that are recorded. It considers the negative accelerations and reports the largest magnitude acceleration. If the current absolute value is greater than the previous maximum value, it will replace the maximum with the current value. These statements are placed within the else statement after the deltaZ value is converted to a string and set to tvZ to be displayed (java code: tvZ.setText(Float.toString(deltaZ)); ). The next lines that directly follow then convert the maximum and current values to strings and set them to the instances of Textview assigned earlier.
/ The following statements will change the value for
* maximum acceleration if the current x, y, or z sensor value
* is higher than the current value of maxX, maxY, or maxZ.*/
if (Math.abs(maxX) < Math.abs(x)) maxX = x;
if (Math.abs(maxY) < Math.abs(y)) maxY = y;
if (Math.abs(maxZ) < Math.abs(z)) maxZ = z;

/* The following code converts the current and maximum
* sensor values into strings and then sets those strings to
* display in the instances of TextView assigned earlier.*/
tvmX.setText(Float.toString(maxX));
tvmY.setText(Float.toString(maxY));
tvmZ.setText(Float.toString(maxZ));
tvlX.setText(Float.toString(x));
tvlY.setText(Float.toString(y));
tvlZ.setText(Float.toString(z));

The second section of code presented is revised within main.xml. This code is
responsible for controlling the display output on the phone. The first six instances of TextView
are associated with the maximum acceleration values while the next six are the current
acceleration values. In each case the name of the variable is displayed first, which is indicated by
“@string/maxX”. The string associated with the value of the variable is displayed second and is
indicated by “@+id/x_max”.

FROM main.xml:

<!--The following group of XML code handles the
display for the maximum acceleration values.-->
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="left"
    android:text="@string/maxX"
    android:textSize="14sp" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/x_max"
    android:gravity="center"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="@string/maxY"
    android:textSize="14sp" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/y_max"
    android:gravity="center"/>
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="right"
The last section of revised code presented is from strings.xml. The app name is changed and the title displayed is also changed. The next strings are also amended to include the units of acceleration. The final six strings are added for the display of maximum and current accelerations in all three axes.

FROM strings.xml:

<!-- The following strings are used for writing to the display. Some of the strings were changed to personalize the app. Others were added to name each number display.-->

-android:text="@string/maxZ"
-android:textSize="14sp" />
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:id="@+id/z_max"
-android:gravity="center"/>
</TextView>

<!--The following group of XML code handles the display for the current acceleration values.-->
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:gravity="left"
-android:text="@string/lastX"
-android:textSize="14sp" />
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:id="@+id/x_last"
-android:gravity="center"/>
</TextView>
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:gravity="center"
-android:text="@string/lastY"
-android:textSize="14sp" />
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:id="@+id/y_last"
-android:gravity="center"/>
</TextView>
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:gravity="right"
-android:text="@string/lastZ"
-android:textSize="14sp" />
<TextView
-android:layout_width="wrap_content"
-android:layout_height="wrap_content"
-android:id="@+id/z_last"
-android:gravity="center"/>
Hello World, MyAccelerometerActivity!

Now the screenshots of the application running on my Android phone are presented. These images are recorded while the phone is connected to the computer using the DDMS within Eclipse. There are two images of slower controlled movements and two images of faster movements. The strings from strings.xml above can clearly be seen in the following screenshots.

![Phone screenshot](image)

The first image was recorded when the phone was lying flat on the table and while the phone was sliding down which is in the positive y-direction. (The coordinate system on my phone is setup so that positive points down for y, to the left for x, and toward the ground or into the page for z). It can be seen that the displayed current acceleration value, which is the gravity acting in the positive z-direction, is close to the handbook value of 9.81 m/s². The maximum acceleration values are also close to the current values since the app had just started and the only motion of the phone was sliding upwards while lying flat. In fact, the maximum acceleration in the x-axis exactly matches the current acceleration.
The second image was taken after the phone was picked up and held in the air. The phone was then moved horizontally to the left in the air. This movement is indicated by the current X-axis acceleration. The maximum x-axis acceleration also shows the largest magnitude acceleration which happens to be negative. Gravity is acting in the positive y-direction, which is down. The maximum acceleration from the phone lying flat on the table is retained and can be seen in the maximum acceleration of the z-axis. The maximum acceleration of the y-axis has also adjusted to show that the phone was lifted and held vertically.

The third image recorded is when shaking the phone rapidly. The acceleration values are all over the place so it cannot be easily inferred which orientation the phone was moving. However, it can be seen that the current acceleration for the y-axis is just below the magnitude of the maximum y-axis acceleration so the maximum was not replaced.
The final screenshot was taken when the phone was held somewhat flat and moved rapidly up and down along the y-axis. The change in y-axis acceleration shows this movement as well as the higher current y-axis acceleration. It can also be seen that the image of vertical arrows is displayed indicating that the y-axis acceleration is greater than the x-axis acceleration. The maximum acceleration in the x-axis also shows how the application will indicate the largest magnitude of acceleration, regardless of sign, so the user will know the direction of that maximum acceleration.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Axis (m/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Y-Axis (m/s)</td>
<td>15.803819</td>
</tr>
<tr>
<td>Z-Axis (m/s)</td>
<td>0.0</td>
</tr>
<tr>
<td>Maximum Acceleration X-Axis (m/s)</td>
<td>-2.6281822</td>
</tr>
<tr>
<td>Maximum Acceleration Y-Axis (m/s)</td>
<td>22.633747</td>
</tr>
<tr>
<td>Maximum Acceleration Z-Axis (m/s)</td>
<td>20.437057</td>
</tr>
<tr>
<td>Current Acceleration X-Axis (m/s)</td>
<td>-1.9221634</td>
</tr>
<tr>
<td>Current Acceleration Y-Axis (m/s)</td>
<td>-13.572403</td>
</tr>
<tr>
<td>Current Acceleration Z-Axis (m/s)</td>
<td>7.963</td>
</tr>
</tbody>
</table>
Copy and paste this test section at the end of your report as an appendix. This section is to be graded. Select (using color, underline or enlarge) the best answer out of the 3 blue-colored choices in each question.

There are total 30 items with questions. Make sure you go through every item.

1. In Eclipse, automatic compilation is often turned on by default. This means that every time you (open, close, Save) a project file, Eclipse recompiles the changes for your application package. You can also manually compile your code if you so desire. You can choose the (clean, rebuild, reset) option that will allow you to do full rebuild of all files.

2. A (method, class, package) provides a definition for an object. You could create different (instances, examples, sub-objects) of the object.

3. When you want to reference an object from within another class, you need to include an (open, input, import) statement in the top of your class file.

4. A class could include some (data fields, inputs, outputs) to describe an object, e.g. species, color and size of a fish(); as well as some of its behavior in the form of (descriptions, packages, methods) like eat(), sleep(), and makeBabyFish().

5. A special type of method, called a (initialization, open, constructor) is used to create and initialize an object; these are named the same as their class and may include parameters.

6. In Java, (subclass, inheritance, transfer) means that Java classes can be organized into hierarchies with lower, more specific, classes in the hierarchy inheriting behavior and traits from higher, more generic, classes. The keyword to define a derived class, or subclass, uses the (extends, subclasses, derives). A method is the subclass can be customized by using the (new, revised, override) mechanism.

7. While a class defines an object, an (library, interface, common) defines some behavior that can be applied to an object. A class can implement an interface using the (implements, extends, interacts) keyword.

8. A (package, superclass, library) is simply a set of classes and interfaces, bundled together. You could use edu.colorado.yourID to define a unique name of your own package.

9. For comments, you can encapsulate any number of lines of text by beginning your comment with (/*…*/, /…../, *…..*). You can also provide comments after code on a single line using (**, /**/, //)

10. Java variables generally fall into two categories: primitive data types, like int, float, double, char, etc. and Java objects as defined by a (package, interface, class) definition.

11. A variable is only valid within its territory, or scope. When a variable is defined, it is valid within ([ ], { }, //).
12. Primitive types variables can be defined by specifying the datatype, followed by the variable name, then an equals sign and an (constant, final, initial) value. All Java statements end with a (period, semicolon, comma).

13. Perhaps the most common object you’ll use in Android applications is the (String, Text, Display) class, which is used to encapsulate human-readable text characters, which are often displayed to the screen.

14. You can control the visibility of a class as well as its variables and methods by specifying an item’s access level. The access levels are: public, protected and private. Generally speaking, if you want something to be accessible from outside a class, use (public, protected and private).

15. Finally, if you want to loop for a specific number of iterations, you can use a for() loop. A for() loop has three parameters: the initial value, the terminating value, and the (incrementing, intermediate, adding) value.

16. For an Android phone, if you push the device on the left side, so it moves to the right, the x acceleration value is (positive, negative, zero). If you push the device on the bottom, so it moves away from you, the y acceleration value is (positive, negative, zero). If you push the device toward the sky with an acceleration of A m/s², the z acceleration value is equal to A (-, +, x) 9.81.

17. The advantage to declaring your user interface in (Java, HTML, XML) is that it enables you to better separate the presentation of your application from the code that controls its behavior. After declaring your application's default layouts, you could then add code in your application that would modify the state of the screen objects, using (View, Display, Print) class, at run time.

```java
// ###############################################################################
package teach.sensing.accelerometer;

import android.app.Activity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener { /* 18. (Activity, MainActivity, SensorEventListener) is a subclass through inheritance. */
    private float mLastX, mLastY, mLastZ;
    private boolean mInitialized;
    private SensorManager mSensorManager;
    private Sensor mAccelerometer;
    private final float NOISE = (float)2.0; /* 20. In the Java programming language, the (private, float, final) keyword is used in several different contexts to define an entity which cannot later be changed. */

    SensorEventListener { //19. (Activity, MainActivity, SensorEventListener) is an interface.
        private float mLastX, mLastY, mLastZ;
        private boolean mInitialized;
        private SensorManager mSensorManager;
        private Sensor mAccelerometer;

        private final float NOISE = (float)2.0; /* 20. In the Java programming language, the (private, float, final) keyword is used in several different contexts to define an entity which cannot later be changed. */
    }
```
/** Called when the activity is first created. */

@override
/** 21. Customize the following (data, subclass, method) in the class by using the override mechanism. */
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mInitialized = false;
    mSensorManager = (SensorManager)
        getSystemService(Context.SENSOR_SERVICE);
    mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    mSensorManager.registerListener(this, mAccelerometer,
        SensorManager.SENSOR_DELAY_NORMAL);
}

protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mAccelerometer,
        SensorManager.SENSOR_DELAY_NORMAL);
}

protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
}

@Override
public void onSensorChanged(SensorEvent event) {
    /* 22. In this case the system invokes the onSensorChanged() (method, data, class) providing
    you with a SensorEvent object. A SensorEvent object contains information about the new sensor data, including: the
    accuracy of the data, the sensor that generated the data, the timestamp at which the data was generated, and the new
    data that the sensor recorded. */

    TextView tvX = (TextView)findViewById(R.id.x_axis); /*  23. tvX is defined as an instance of a TextView
    (package, method, class), which displays text to the user using android:id. */
    TextView tvY = (TextView)findViewById(R.id.y_axis);
    TextView tvZ = (TextView)findViewById(R.id.z_axis);
    ImageView iv = (ImageView)findViewById(R.id.image);

    float x = event.values[0]; //24. Acceleration force, in m/s^2, along the (x, y, x) axis, including gravity.
    float y = event.values[1]; //25. Acceleration force along the (x,y,x) axis, including gravity.
    float z = event.values[2]; //26. Acceleration force along the (x,y,z) axis, including gravity.

    if (!mInitialized) {
        mLastX = x;
        mLastY = y;
        mLastZ = z;
    }
tvX.setText("0.0");
tvY.setText("0.0");
tvZ.setText("0.0");

mInitialized = true;

} else {

float deltaX = Math.abs(mLastX - x);
float deltaY = Math.abs(mLastY - y);
float deltaZ = Math.abs(mLastZ - z);

if (deltaX < NOISE) deltaX = (float)0.0;
if (deltaY < NOISE) deltaY = (float)0.0;
if (deltaZ < NOISE) deltaZ = (float)0.0;

mLastX = x;
mLastY = y;
mLastZ = z;

tvX.setText(Float.toString(deltaX));
tvY.setText(Float.toString(deltaY));
tvZ.setText(Float.toString(deltaZ));

iv.setVisibility(View.VISIBLE);
if (deltaX > deltaY) {
iv.setImageResource(R.drawable.horizontal);
} else if (deltaY > deltaX) {
iv.setImageResource(R.drawable.vertical);
} else {
iv.setVisibility(View.INVISIBLE);
}

}

public void onAccuracyChanged(Sensor sensor, int accuracy) {

// TODO Auto-generated method stub

}


//##########################################################

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
<TextView
    android:paddingTop="20dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
    android:textStyle="bold"
    android:gravity="center"
    android:text="@string/title"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="left"
    android:text="@string/axisX"
    android:textSize="14sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/x_axis"
    android:gravity="center"
    *
    28. The android:id supplies an identifier name for this view, to later retrieve it with View(findViewById(), View(), Find()) *
    *
    29. android:id="(@+id/my_id", "@+id", "@+my_id") which allows you to later retrieve the view with *
    findViewById(R.id.my_id). *
    android:gravity="center"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="@string/axisY"
    android:textSize="14sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/y_axis"
    android:gravity="center"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/z_axis"
    android:gravity="center"/>

<ImageView
    android:paddingTop="10dip"
    android:layout_width="wrap_content
    android:layout_height="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/image"
android:layout_gravity="center"
android:visibility="invisible"/>

</LinearLayout>

// #####################################################

<?xml version="1.0" encoding="utf-8"?>
<resources>

<string name="hello">Hello World, MyAccelerometerActivity!</string>
<string name="app_name">My Accelerometer</string>
<string name="title">Shaker</string>
<string name="axisX">X-Axis</string>
<string name="axisY">Y-Axis</string>
<string name="axisZ">Z-Axis</string>

/* 30. Remove the above line and compile the program using File → Save or Project → Clean. You would expect to see an error message beginning with No source found, No problem found, No error found. */

</resources>